Universidad de Medellín

A Representation Proposal of Practices for Teaching and Learning Software Engineering Using a Semat Kernel Extension*

María Clara Gómez Álvarez[**]
Rubén Sánchez-Dams[***]
Álvaro Alexander Barón Salazar[****]

Received: 16/11/2016 • Accepted: 05/06/2017
DOI: 10.22395/rium.v17n32a7

Abstract

Software engineering is a discipline oriented to the definition of methods, techniques and tools for developing software products in an efficient and rapid way. Growing demand of such products generates the need of a large amount of software engineers with the technical and social competencies required by software industry. This situation is a challenge for Higher Education Institutions in terms of a training process of future professionals of this discipline. In this sense, such institutions are exploring active teaching strategies for promoting the needed competencies in students. However, an integrated proposal of these teaching approaches is still underdeveloped. In this paper, the authors present a proposal for representing practices for teaching and learning software engineering, oriented to identify the main concepts included in any type of these practices. The proposal is based on the Semat kernel –Essence standard– as universal framework for representing software engineering practices, defining an extension to such kernel. Finally, we present a representation example of a software engineering teaching and learning practice using the Semat Kernel Extension proposed.

Keywords: Semat Kernel, practice, software engineering teaching and learning

* Article of ongoing investigation, titled: *Formulación de una teoría general para la enseñanza de ingeniería de software*. Execution time: 2 años. Financing entity: Universidad Nacional sede Medellín.

** Systems and Informatics Engineer, Ph. D. Universidad Nacional de Colombia, sede Medellín, Facultad de Minas. Medellín, Colombia. Research group Arkadius, Universidad de Medellín. mcgomez@udem.edu.co

*** Electronic Engineer, Ph. D. Universidad Nacional de Colombia, sede Medellín, Facultad de Minas. Medellín, Colombia. Research group Lenguajes Computacionales. rubendams@gmail.com

**** Systems Engineer, Ph. D. (c). Universidad Nacional de Colombia sede Medellín, Facultad de Minas. Medellín, Colombia. Research Group Galeras.Net, Universidad de Nariño. abaron_98@udenar.edu.co

Propuesta de representación de prácticas de enseñanza y aprendizaje de ingeniería de software usando una extensión del núcleo de Semat

Resumen

La ingeniería de software es una disciplina orientada a la definición de métodos, técnicas y herramientas para el desarrollo eficiente de productos de software. La demanda creciente de estos productos genera la necesidad de contar con una gran cantidad de ingenieros de software con las competencias técnicas y sociales requeridas por la industria. Esta situación es un desafio para las instituciones de educación superior en relación con el proceso de enseñanza de los futuros profesionales de esta disciplina. En este sentido, estas instituciones están explorando estrategias activas de enseñanza para promover en los estudiantes las competencias necesarias. Sin embargo, una propuesta integradora de estos enfoques de enseñanza no ha sido desarrollada hasta ahora. En este artículo los autores describen una propuesta para representar prácticas de enseñanza-aprendizaje de ingeniería de software, orientada a identificar los principales conceptos incluidos en cualquier tipo de práctica. Esta propuesta esta basada en el núcleo de Semat, del estandar Essence, como marco de trabajo universal para la representación de prácticas de ingeniería de software, definiendo una extensión de dicho núcleo. Finalmente, presentamos un ejemplo de representación de una práctica de enseñanza-aprendizaje de ingeniería de software usando la extensión del núcleo de Semat propuesta.

Palabras clave: nucleo de semat, práctica, enseñanza-aprendizaje de ingeniería de software.

Proposta de representação de práticas de ensino-aprendizagem de engenharia de software que usam uma extensão do núcleo da Semat

Resumo

A computação em nuvem é um modelo onipresente que permite o fornecimento de serviços a clientes que têm acesso a ela de forma fácil e rápida. O software como serviço (SaaS) é um dos modelos de maior uso, por meio do qual os aplicativos se estendem e armazenam pelos clientes via internet, com um navegador web pago por uso. Contudo, por sua complexidade e características — reuso, escalabilidade, elasticidade e personalização —, o SaaS é definido por fluxos de trabalho compostos de microsserviços ou serviços atômicos alojados geograficamente em diferentes lugares. Nesse conteto, o SaaS pode apresentar comportamentos anormais nos resultados ou falhas na aplicação final do usuário em tempo de execução. Neste artigo, apresenta-se um modelo de orquestração dinâmica, cujo objetivo é diminuir as falhas ou os comportamentos anormais dos serviços que participam do processo de execução dos aplicativos de negócios.

Palavras-chave: computação em nuvem; fluxos de trabalho; microsserviços; orquestração dinâmica; serviços atômicos; software como serviço.

INTRODUCTION

Software products have an increasing demand in sectors like banking, government, assurance, among others. Such situation intensifies the need of software engineers with competencies oriented to software quality and shorter delivery times [1]. This implies that Higher Education Institutions should simultaneously promote: a) assimilation of techniques of software engineering for activities like project estimation, software testing, and requirements elicitation, among others; and b) social competencies for teamwork like negotiation, effective communication and leadership [2].

In this context, such institutions are exploring active teaching strategies, besides lectures and class projects. The purpose of this exploration is promoting in students the expected competencies for software industry to meet its needs. Some active strategies used in software engineering teaching are collaborative learning, project-based learning, studio-based learning and simulation [3-6]. One of the contributions of such strategies is increasing the motivation and commitment levels in students that face their learning process [7].

However, although there are different teaching experiences for software engineering, each professor incorporates such strategies in a particular manner and an integrated proposal of these experiences is still underdeveloped. These approaches are often called Practices For Teaching And Learning Software Engineering (PTLSE for its acronym) in the literature.

In this sense, the authors propose using the Essence standard as a common representation resource of any PTLSE [8]. In order to achieve this aim, the authors present an extension to the Semat kernel considering elements of the PTLSE domain. The proposal is preliminary; this is an intermediate result of a work in progress. At present, the proposal lacks a formal validation. However, the authors expose the use of the extension to represent a new PTLSE, showing an exemplification like an alternative validation. Also, this paper describes a methodology for defining a Semat Kernel Extension.

The paper is organized as follows: in Section 1 we present the main concepts underlying this proposal like software engineering, teaching and learning practices and an overview of the Semat initiative. Then, in Section 2 some experiences of software engineering teaching, analyzed for generating the extension are presented. Later, Section 3 describes the methodologies used by the authors for generating the Semat Kernel Extension. Section 4 presents the results in terms of the Semat Kernel Extension proposed and an example of a PTLSE representation using that extension. Finally, in Section 5 the conclusions and possible future work are presented.

1. THEORETICAL FRAMEWORK

1.1. Software Engineering

Software Engineering is a discipline comprising all the dimensions related to software development, from early stages of specification of system requirements, to later maintenance and implementation in a production environment [1]. Software engineering can be defined as the application of a systematic, disciplined and quantifiable approach to software development [9]. Such application comprises the development of methods, tools, and theories for supporting software production, and the management of projects, involving people, processes, and technological tools.

Software engineering differs in several aspects from traditional engineering due to special features of a software product. In this context, concepts like abstraction, modeling, organization and representation of information, and change management are important [10]. So, the application of software engineering methods, techniques, and principles deals with technical issues of computer science and soft competencies like project management, communication, negotiation, and teamwork, among others.

1.2. The Practice Concept

The concept of practice is common in all engineering areas [11-16]. Examples of proposals of a definition of practice in different areas are listed below.

Madigan *et al.* define practice in the context of civil engineering as a way to perform an activity in a determined process. Practice allows high levels of product and process quality. In addition, it includes control, measurement and assessment mechanisms in coherence with environment and life quality. In civil engineering, practice is presented as a standard, rule or law [11].

In electric engineering, Basso presents practice as a manner to perform a work in a project with features of innovation and guaranteed quality. Practice includes a guide, risk control, monitoring and measurement [12].

In a standard for chemical engineering of the Government of Alberta, practice is defined as a valid strategy to perform a task. Practice is implemented, maintained and evaluated with quality criteria. Practice includes: risk control, rigorous assessment and measurement mechanisms [13].

In process engineering context, Dhole defines practice as a successful way to perform an activity in a process. Practice includes development and innovation strategies [14].

In software engineering, Capability Maturity Model Integration for Development (CMMI-DEV) defines practice as a set of process areas to satisfy a set of goals that can be generic or specific [15].

OMG (Object Management Group) in its standard: Essence Kernel and Language for Software Engineering Methods –also known as Semat kernel–, defines software practice as a repeatable approach for a specific purpose. Practice has a clear goal expressed in terms of results that can be applied. Software practice provides a guideline to help practitioners on what should be done to achieve the goal, to ensure such goal is understood and to verify what is achieved [16].

1.3. Teaching and Learning Practices

Teaching and learning practices include a set of guidelines and activities used by professors to prepare prospective professionals and promote the development of technical and social competencies in students [17]. In general, a teaching and learning practice has six main components shown in figure 1.

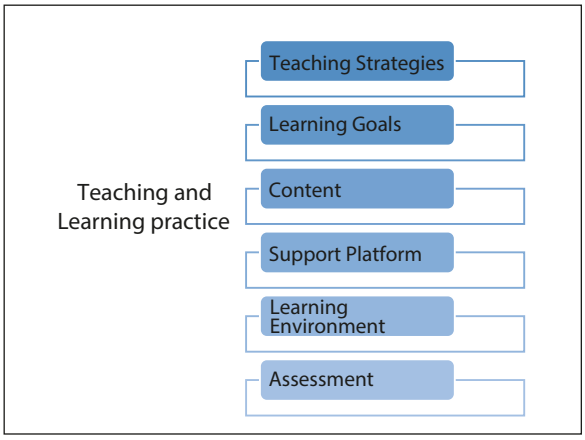

Figure 1. Teaching and learning practice components

Source: authors

Teaching strategies

Set of resources, techniques and procedures used by professors for generating meaningful learning in their students [18]. A systematic process of planning, design, implementation and evaluation of teaching and learning activities is one of the ways for achieving such meaningful learning.

Learning goals

Learning goals refers to the cognitive activities that students should be able to do at the end of a teaching strategy application. Such goals are related to knowledge,

abilities and attitudes expected by professor in a curriculum planning and previous courses [19].

Content

Content is a set of subjects characterizing a phenomenon or event. In a course context, content is a set of specific subjects described in a syllabus and that correspond to the knowledge shared by the professor with their students for developing technical competencies.

Support platform

Support platform includes a set of technological tools for complementing the teaching-learning process. Such tools allow students to learn and work by themselves, using the resources and activities available in these applications [20-21].

Learning environment

Learning environment refers to the different contexts, physical locations and conditions in which students learn [22].

Assessment

Assessment is a key component of the teaching-learning process [23]. The assessment purpose is to verify if students develop competencies expected after the application of teaching practices. Such activity is traditionally carried out via written exams, but other evaluation methods are appearing, like group activities or ludic workshops [24-25].

1.4. Theory and conceptualization

In this section, we explain the relationship between the creation of theory and the activity of conceptualizing a particular domain.

Conceptualization

According to the Oxford dictionary, conceptualization is “the action or process of forming a concept or idea of something” [26]. Conceptualization is a process of abstraction to identify concepts in a disciplinary domain. This process is achieved through categorization and definition of classes of elements within a disciplinary domain under study. Such categorization achieves a typification constituting a concept when being named. Conceptualization is indispensable when representing phenomena of a disciplinary domain. Thus, conceptualization is related to any representation proposal of practices because concepts are the elements used in a representation [27].

Pre-conceptual schema

A pre-conceptual schema is a diagram used to represent knowledge of any disciplinary domain unambiguously, using controlled graphical language. The graphical language is a set of symbols representing: a) concepts; b) structural relationships between concepts; c) dynamic relationships denoting actions or operations; d) connectors to relate the rest of the language elements; and e) implications of cause-effect relationships. Figure 2 shows the symbols of these elements of the pre-conceptual schema.

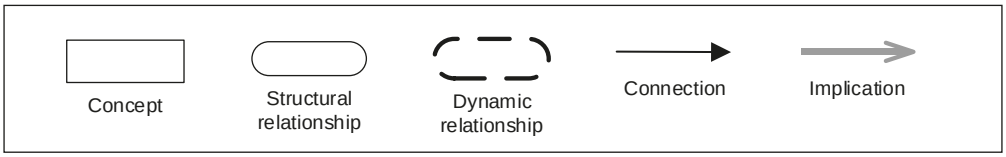

Figure 2. Pre-conceptual schema main elements

Source: authors

Also, executable pre-conceptual schemas allow the instantiation of a conceptual model. In such schema, some symbols can be assigned values. In this way, it is possible to represent one or several situations of a phenomenon or disciplinary domain [28-29].

Pre-conceptual schema is useful to do conceptualizations, because this has important features. First, this representation has proximity to both the natural language and the formal logic. Second, schemas can represent knowledge about any domain. And third, using pre-conceptual schema, you can define operations among concepts.

Theory

A theory is a common conceptual framework for describing a phenomenon occurring in a disciplinary domain [30]. Researchers should make abstractions about a disciplinary domain to describe a discipline. For humanity, phenomena and concepts of a domain are diffuse and complex before the creation of theories. Abstraction is the process of identifying only the necessary and sufficient elements to describe a disciplinary domain. When researchers start studying a disciplinary domain, they identify concepts, some concepts are essential within the domain. Such concepts are called constructs. In addition, researchers identify interactions between constructs; these are called propositions. So theories are defined by constructs and propositions being stated by equations, statements, assumptions or laws [30-31].

Theory validation

Validation is achieved through experimentation, by making use of proposed propositions in a disciplinary domain. Any proposition can be falsified. When results of

experimenting with a proposition gives way to errors, faults, or inaccuracies, a need arises to expand a disciplinary domain. Expansion of a disciplinary domain is done by: a) adjustment of existing propositions, according to new discoveries; or b) searching for new propositions to explain new observations found in experimentation. Cases where a proposition fails are important, because such cases delimit the scope of the theory. A failure helps to increase understanding of a disciplinary domain, delimiting proposition validity, its level of generality, and abstraction level where a proposition is applicable [27].

1.5 Essence standard like a theoretical framework for practices representation

Semat is a community of people, some companies, and some universities around the world, supporting an initiative to create a common ground, a kernel or a foundation for software engineering [32]. Semat community provides Essence for creation, use and improvement of software engineering methods. Essence is a means for representing software engineering development processes. Essence is a standard adopted by OMG (Object Management Group) in June 2014 [16].

Development processes address any approach in software engineering. When users of a software development approach work in a particular and repetitive way, and the approach addresses several dimensions of software development, it is said to be a development method. Before Essence, users represented their methods with other standards [33-35]. For these representations, users make their methods mainly with activities and other elements. In this way, however, activities are diluted in the method, addressing any dimension of development approach [36]. Users can more easily adjust or implement methods using Essence. If users practice does not work, user can change to other with the same objective, without changing the whole method. Similarly, a user can add a new practice to his method, addressing a new dimension or goal [37].

Essence consists of a kernel and language. The kernel is a set of concepts and their relationships. Such concepts and relationships are essential and always present in any software development process. The language is a set of elements and rules of expression to represent development process.

Overview of Semat kernel specification

For theorizing software engineering, the kernel is important because it explicitly contains constructs and propositions of this disciplinary domain. Thus, the standard is a theoretical proposal; it contains the representation mechanisms of the software engineering development process [38].

The kernel elements are organized into three areas of concern: customer, solution, and endeavor. The areas "customer" and "solution" are related to problem and solution domain, respectively. The area "endeavor" is about the resources involved in the solution development. The kernel main constructs are "alphas", "activity spaces", and "competencies". Each area of concern contains a small number of such constructs.

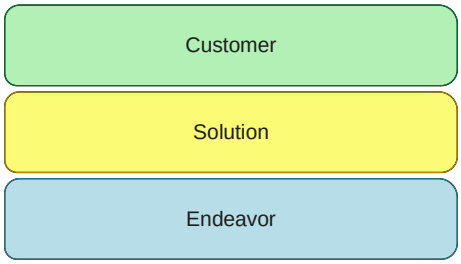

Figure 3. Areas of concern of the kernel

Source: [16]

Figure 3 shows the areas of concern of the Semat kernel. Alphas representations of the essential things with which everyone works in an endeavor. There are seven alphas with which progress and health of the most important elements of software engineering are measured. The alphas and their relationships are presented in figure 4.

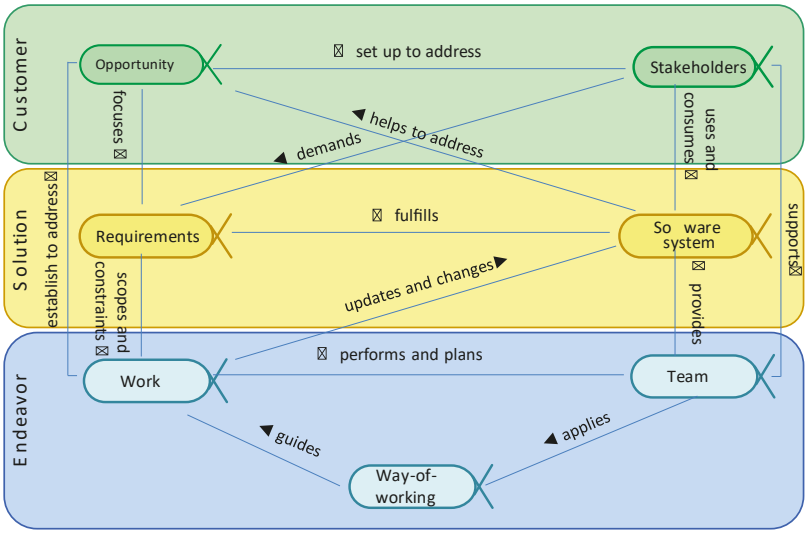

Figure 4. Alphas and their relationships

Source: [16]

Activity spaces are representations of essential things to do, what is done. Activity spaces provide descriptions of challenges that a team faces for the development,

maintenance and support of software systems. Team follows such description and does activities inside of space activities to meet these challenges. They are presented in figure 5.

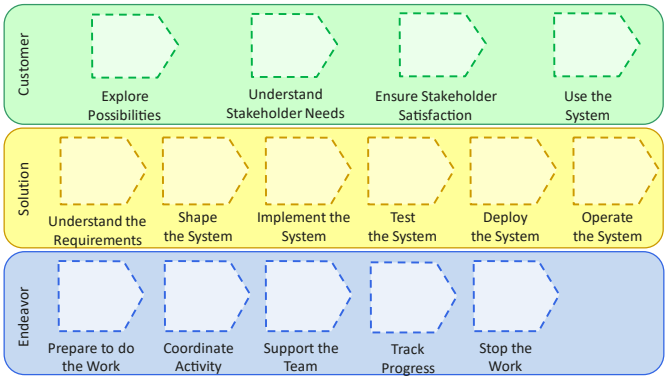

Figure 5. Activity spaces

Source: [16]

Competencies are the capacities to perform work. Competencies are representations of the key skills required to carry out software engineering. They are presented in figure 6.

Figure 6. Competencies

Source: [16]

Language has an extension mechanism to the kernel elements, with which users can represent the peculiarities of their development processes. For example, a user may propose sub-alphas, competencies, or activities relating to space activities for a particular development practice. The authors recommend readers as Essence introductory material a summary presented in [39], for the use of Essence the book [37], and as a reference document, the standard itself [16].

Overview of Semat Language Specification

Figure 7 illustrates a conceptual overview of the Essence language, the main elements of the language and their most important associations are showed. The elements centered in the figure are used to describe the contents of a kernel. They provide the abstract and essential thing to do (alpha), things to work with (activity space), and things to

know (competency) in software engineering endeavors. These elements are sufficient to know about the state, progress, and health of a software engineering endeavor.

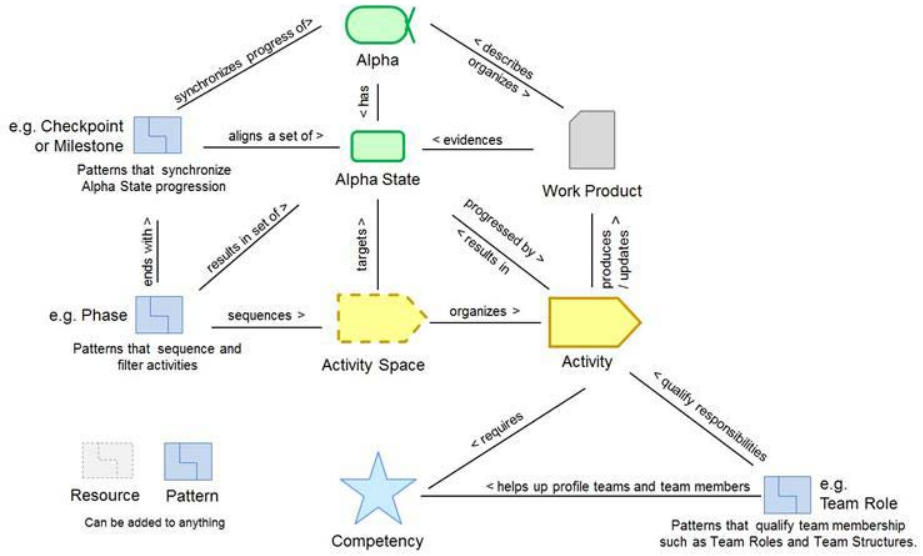

Figure 7. Conceptual overview of the essence language

Source: [16]

Elements used in a kernel represent abstract elements; concrete guidance can be created through practices. Practices allows adding elements like patterns and resources. Patterns and resources are generic concepts that can be attached to any language element. They are not considered by the dynamic semantics of the language. Work products are the concrete things to work with, and they are evidence for the states about an alpha. Activities are explicit guidance on how to produce or update work products.

Alpha states and activities describe the dynamic semantics of the language. An endeavor is oriented toward the state a team wants to reach, activities drive the endeavor towards that goal.

2. BACKGROUND

In this section, the authors present some approaches about software engineering teaching and learning practices. These include approaches like prototype-development; project and studios based on learning, besides of lectures and class projects.

Qureshi *et al*. propose a model for teaching software project management that seeks to reduce the failure rate in this type of projects [3]. Such model has a strong

emphasis in hand-on practice and creates an atmosphere as a tool for motivating students about software projects management. The components of this model are: a) the instructor, responsible for monitoring the software project; b) the students working in groups for developing the software product; and c) software industry acting as a client of the software in construction. Finally, the teaching method associated to this model includes the examination of the software product progress and of other projects activities set by the instructor; the development of a prototype by student groups and constant feedback from the instructor and clients to students.

Furthermore, Ma *et al*., present a prototype-based teaching model for computer education incorporating innovation as a fundamental ability to promote in students [4]. Under the prototyping, such teaching process is divided into classroom activities and extra-curricular learning websites. Extracurricular websites imply that students prepare each lesson before class. Meanwhile, Muller *et al*., describe the use of a practice-oriented education for software engineering where students collaboratively develop a software system over two consecutive semesters [6]. The learning goals of this proposal are: a) autonomous development of a viable solution for a given problem; b) project planning and management; c) quality assurance within the project; and d) effective communication inside the team and with the client. Such proposal deals with theory and practice and includes teaching activities like presenting the concepts, the integrated design practices in software engineering, assessment and constant feedback for students, among others.

In this practical context, Bull and Whittle propose a studio-based approach for supporting reflective practice in software engineering [5]. Such proposal promotes reflection-in-action, or reflection during the problem-solving process. This implies that students develop a software product for resolving specific problems and later consider what could have been done differently the next time. A studio is a room for supporting collaborative and project-based learning where professors act as coaches and favor the process of learning-by-doing.

Another example of practical teaching system for software engineering is the proposal of Guowei *et al*., composed of three dimensions that aim to improve the software development competencies and innovation sense of students [40]. The first dimension is “interest driven” and includes the students’ development of simple videogames as a motivation to learn. The second dimension is “contest driven” oriented to practices for promoting competencies development like students’ self-learning, teamwork and cooperative learning. The third dimension is “project driven” or project-based learning. Such dimension promotes innovation inviting students to participate in research projects.

The revision of previous work includes the construction of a pre-conceptual schema for each proposal of software engineering teaching practice. The goal of this analysis is identifying the main concepts (constructs) and relationships (propositions) of these PTLSE. As an example, the pre-conceptual schema for the proposal of Bull and Whittle is shown in figure 8 [5].

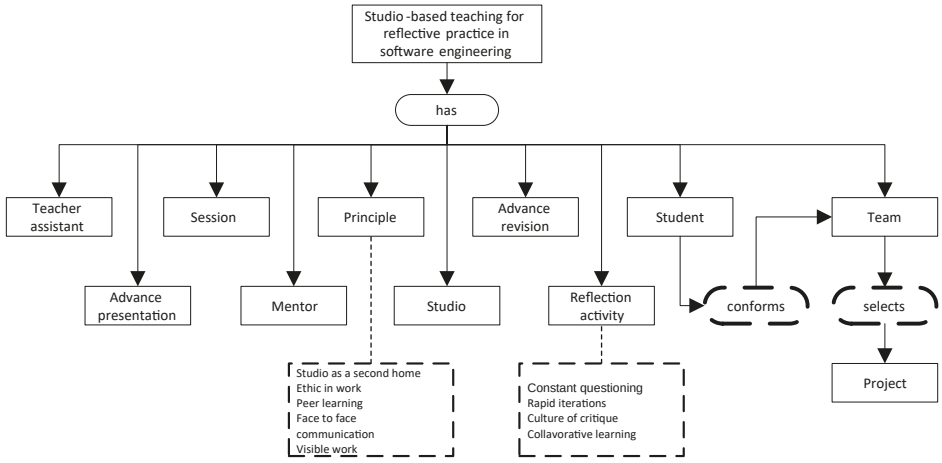

Figure 8. Pre-conceptual schema summarizing Bull and Whittle's teaching and learning proposal for software engineering

Source: authors

3. METHODOLOGY

3.1 *Methodology for Semat kernel Extension Definition*

The methodology for extending the Semat kernel includes two approaches: a) *theory construction* proposed by Sjøberg, Dyba, Anda and Hannay [41], and Ekstedt [42]; and b) *traditional scientific methodology* followed to achieve the steps of the first approach. Both approaches conform a three-phase methodology: definition, construction and validation. In figure 9, we present the relation between approaches' activities and the phases of the methodology.

Moreover, this methodology is based on two hypotheses: a) there exists a set of essential and common constructs and propositions to all PTLSE; and b) such constructs and propositions could be discovered in a finite subset of approaches available in literature.

Definition Phase

The systematic literature review was performed using the guidelines shown in [43]. The research question was: What (model/techniques/practices) are used to teach software

engineering? The research keywords were “teach” or “teaching” AND “software engineering”. The selected sources were ACM, Citeseer, EBSCO, IEEE, Science direct and Scopus. The initial results were composed by 3,896 items and after the application of inclusion and exclusion criteria, defined by this research, the results were 147. Figure 10 shows the activities included in the definition phase.

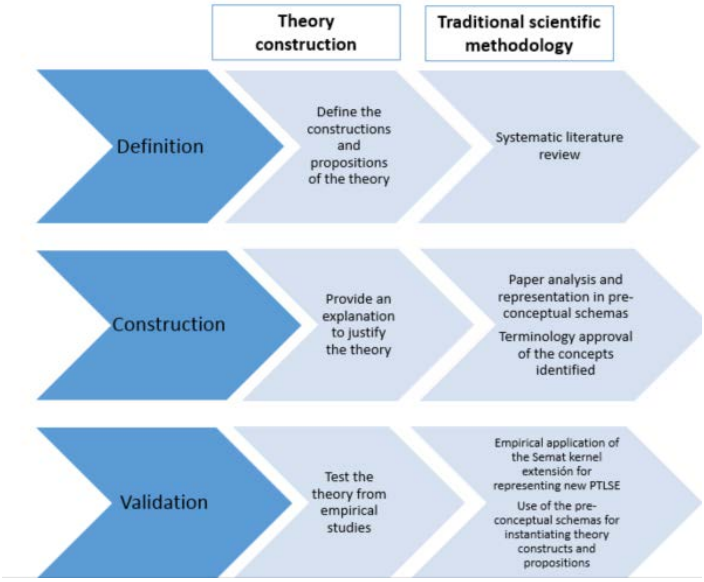

Figure 9. Methodology to generate the Semat Kernel Extension

Source: authors

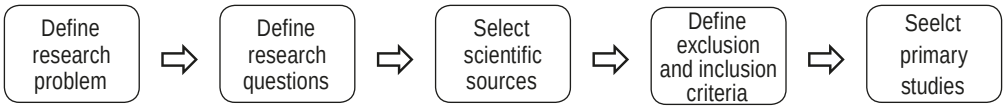

Figure 10. Definition phase activities

Source: authors

Construction Phase

After completing the information extraction of the papers, the authors used pre-conceptual schemas to express the main concepts and relationships identified in a controlled speech. The purpose of this activity was to avoid ambiguities in the concepts identified. Then, a terminological homologation to extract common concepts and relationships followed, including these activities:

1. Identify common concepts in the papers. Some common concepts are: professor, student, stakeholder, learning environment, assessment, learning goal and learning activity.

2. Identify synonymous concepts between the papers and select a unique concept. As an example, the synonymous concepts; traditional activity, learning activity, students' activity, exercise, collaborative activity, reading and activity are synthetized as *learning activity.*

3. Identify relationships between concepts through sentences in the form concept-relationship-concept. Examples: professor-presents-case study or team-implements-software product.

4. Analyze such sentences and eliminate redundancies.

5. Identify concepts without dynamic and structural relationships and return to the papers to identify the missing relationships.

Later, the scope of the extension for representing PTLSE was defined, based on the Semat kernel. Scoping is done with the classification of constructs–concepts–and propositions–relations–extracted from the terminological homologation, which meet the following criteria: a) extend the Semat Kernel; b) orthogonality of constructs; c) generality; and d) establishment of states, checklists and descriptions of concepts by analogies to the Semat kernel.

Figure 11 presents the activities included in the construction phase. From identification of common concepts onwards, the figure shows the activities to reach a terminological homologation of concepts and relationships in the domain of teaching and learning software engineering.

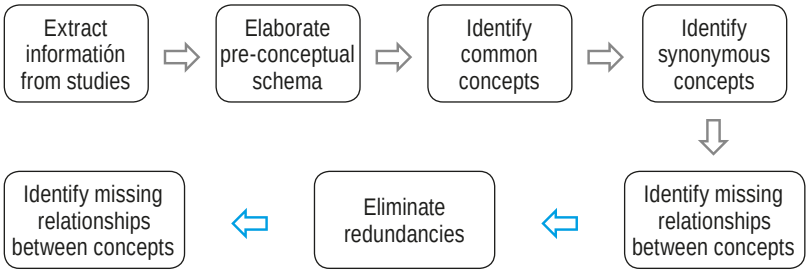

Figure 11. Construction phase activities

Source: authors

Validation Phase

This section describes the proposed validation for the work in progress. At the time of writing, the authors did not apply the formal validation proposed.

The first option for validation is to use of the Semat Kernel Extension to represent PTLSE without including it in the extension definition. One example of such representation is shown in Section 4.2 of this paper. Such example is a case study presented as an alternative to validation.

The second option consists on using executable pre-conceptual schemas for representing the Semat kernel [38]. Figure 12 shows a pre-conceptual schema for Semat initiative understood as a general theory about software engineering. Then, we can define a set of constructs and propositions about Semat extension for representing PTLSE according to such schema. We show four propositions related to alphas Software System and Learning Environment in table 1. Thus, validation consists in verifying the compliance of the suggested propositions for the extension, when analyzing a corpus of PTLSE descriptive documents.

To deepen in the methodology to extend the Semat kernel toward other disciplinary domains, we recommend [44].

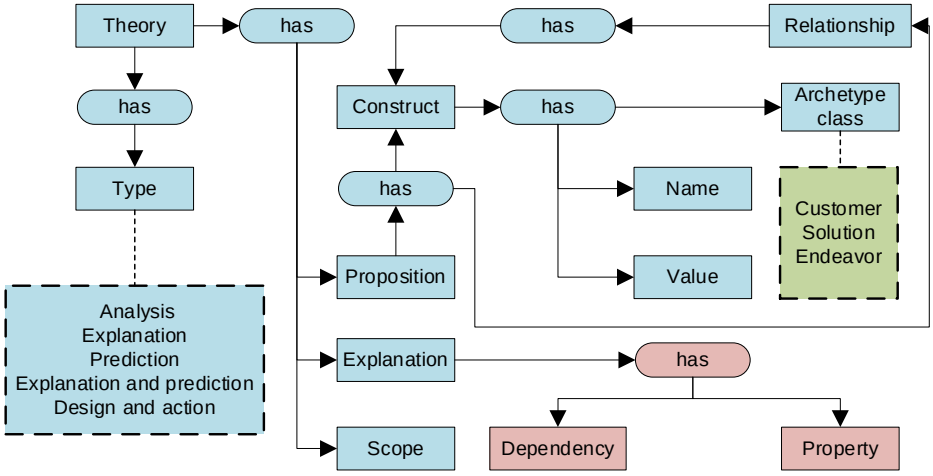

Figure 12. Theoretical pre-conceptual schema for Semat initiative

Source: [38]

The activities included for the first option are shown in figure 17 on this paper. For now, the activities for the second option of validation are showed in figure 13.

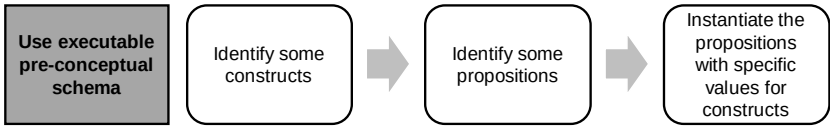

Figure 13. Validation phase activities

Source: authors

Table 1. Some of the suggested propositions for Semat extension

Theory			
Proposition			
Construct			*Relationship*
Name	*Value*	*Archetype Class*	*Name*
Alpha	Team	Endeavor	Participates
Alpha	Learning environment	Solution	
Alpha	Software system	Solution	Object of study
Alpha	Learning environment	Solution	
Alpha	Software system	Solution	Fullfills
Alpha	Requirements	Solution	
Alpha	Work	Endeavor	Updates and changes
Alpha	Software system	Solution	

Source: authors

4. RESULTS

4.1 Semat Kernel Extension for PTLSE

In this section, the authors present a Semat Kernel Extension to represent PTLSE; the authors proposed a similar extension in [44]. In PTLSE, the authors propose to add some sub-alphas and alphas, and one space activity.

Authors identify the lack of an alpha related to teaching-learning process in Semat kernel. Accordingly, and as described in the construction phase of the methodology, authors propose adding the alpha "Learning environment" to the area concerning "solution". Such alpha is an environment in which students develop competencies and tutors guide their activity. It includes teaching style, student types, relationships established in classroom, space, and rules for the use of such space. This alpha also contains the sub-alphas "learning goal", and "learning content".

In the area concerning "endeavor", authors propose adding the sub-alpha "learning activity" to the work alpha, and the sub-alpha "teaching strategy" to the way-of-working alpha. From the papers analyzed, authors describe states, checklists, and descriptive cards of the proposed alpha and sub-alphas. Figure 14 shows the relationships between the proposed alpha and the original alphas of Semat kernel. Figure 15 shows the alphas and their sub-alphas of the extension. Finally, table 2 shows a consolidated of the alpha and sub-alphas.

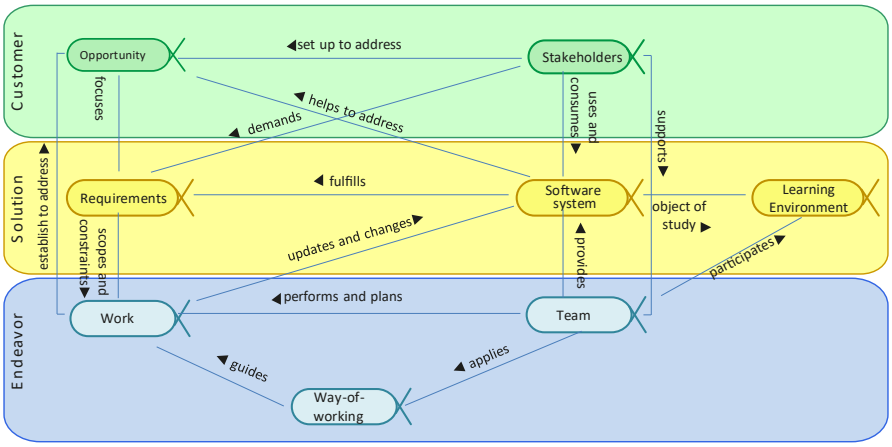

Figure 14. Relationship between the proposed alpha and the Semat alphas

Source: authors

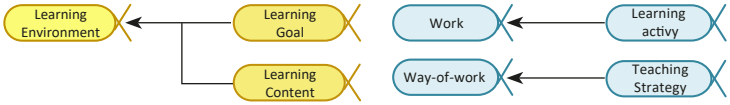

Figure 15. Sub-alphas of the proposed extension

Source: authors

Table 2. Alphas and subalphas proposed and their states

Alpha name	*Sub-alpha*	*Description*	*States*
Learning environment		A space for developing learning activities	– Conceived – Bounded – Designed – In action – Feedback – Finalized
Learning goal	x	Learning objectives correspond to what the student should be able to demonstrate at the end of a teaching and learning process	– Defined – Presented – Addressed – Assessed – Completed
Learning content	x	Set of concepts that are addressed in a training process	– Defined – Bounded – Ready – Presented – Assessed

Alpha name	*Sub-alpha*	*Description*	*States*
Learning activity	x	Actions or tasks performed by the student to learn content and develop competencies	– Conceived – Designed – Made – Completed – Assessed – feedback

Source: authors

We propose a new space activity named "Do learning activity" in the area of concern "endeavor". Such space is a container of the proposed activities for the student. The other activity spaces of Semat kernel meet the needs of PTLSE. In figure 16 the new activity space is shown.

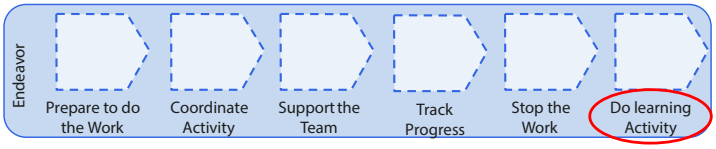

Figure 16. Activity space "Do learning activity"

Source: authors

4.2 REPRESENTATION OF PRACTICES FOR TEACHING AND LEARNING SOFTWARE ENGINEERING

In this section, the authors present the methodology for using Semat extension to represent PTLSE and an application case. Such experience is not included in the systematic literature review for the defined such extension.

Methodology for using Semat extension to represent PTLSE

The activities included in the methodology to apply the Semat extension for representing PTLSE are shown in figure 17.

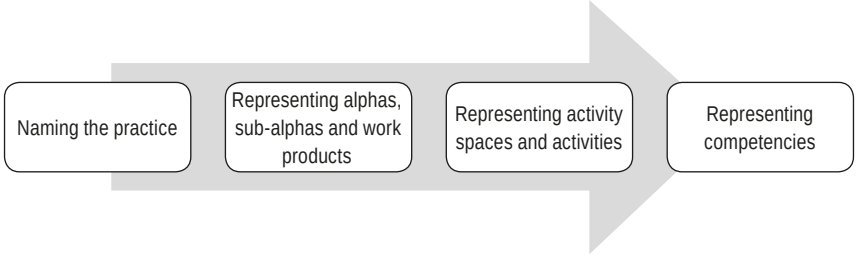

Figure 17. Methodology for using Semat extension to represent PTLSE

Source: authors

1. Naming the practice: you must read the practice description document and make a summary. According to the summary, you must name the practice using the following structure: a nominalized verb to describe the actions of the practice; an adjective to describe the way to perform the practice and an object on which the practice is carried out. For example: "modeling" + "visual" + "software system"..

2. Representing alphas, sub-alphas and work products: you must identify main concepts to be monitored during the PTLSE, to be adapted and represented as alphas and sub-alphas. The "work products" used or generated in the PTLSE are associated with the appropriate alpha or sub-alpha. Then you describe how to use the alphas, sub-alphas, and work products. To obtain the first version of the practices you group alphas, sub-alphas, and work products per areas of interest of the kernel and their relationships.

3. Representing activity spaces and activities: to achieve progress in alpha states, you need to identify the actions performed by professors and students. You identify the key actions of the teaching-learning process as Semat "activities", which are associated with a "space activity". Then you describe the activities and identify the work products required or produced by an activity and that are associated to it. Then, you group the activities and areas of activity in practice, according to areas of interest and activity relations. You also need to adjust the shape of the practice according to what has been done with the alphas. If another practice is necessary to complement progress in an alpha, you will use a space activity, indicating that it should be associated with some activities of a complementary practice. To graph the practice, you should use the standard notation for the corresponding diagrams.

4. Representing competencies: identify competencies necessary to efficiently perform the practice and represent it using the Semat Kernel Extension.

Application case: Collaborative Software Engineering Learning Environment Associating Artifacts Management with Communication Support

A summary about this practice is:

> […] most software development is conducted by organizing a team. Several communications are exchanged there. Software development is often run under a time and/or geographically distributed environment, such as global software development or open source software development in these days. In addition, several intermediate artifacts are created in software development. Project-Based Learning (PBL) that has the abovementioned features has been adopted in edu-

> cational institutes such as universities. Artifacts and communication messages should be recorded in a systematic way and they are required to manage for reference. To realize this requirement, it is indispensable to construct a support system. Some communications exchanged during software development are closely related to various types of artifacts. Therefore, this aspect should be supported by a system. This practice proposes a learning environment for collaborative software development that associates artifacts management with communication management [45].

The following describes the application of the activities presented in the previous section for representing PTLSE with the Semat extension proposed.

1. Naming the practice: according to the above and the methodology, the name of the practice is Collaborative Software Engineering Learning (CSEL). Learning is a nominalized verb that represents the set of practice actions. Collaborative describes the approach with which the practice is carried out. Software system is the object on which the practice is carried out

2. Representing alphas, sub-alphas and work products of CSEL practice: CSEL practice proposes a learning environment for collaborative software development that associates artifacts management with communication management. CSEL practice, alphas, sub-alphas, work products and its relationships are shown in figure 18.

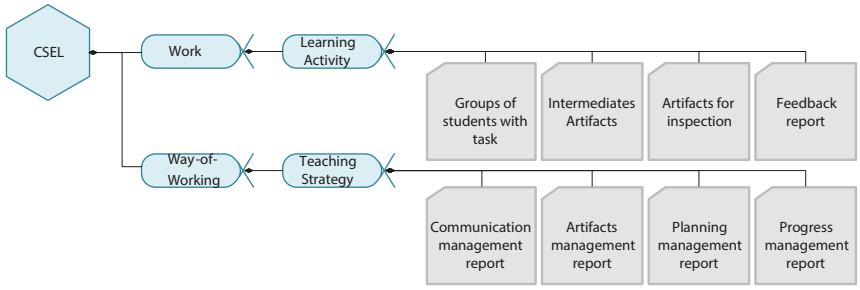

Figure 18. Alphas, sub-alphas and work products of CSEL practice

Source: authors

Each of the sub-alphas and alphas that are part of the extension are instantiated in elements of the CSEL proposal. This is a way of demonstrating that the elements that make part of a PTLSE can be represented by elements that are part of the kernel and the Semat extension. Thus, it is possible to exploit the advantages of the Semat kernel to monitor the health and progress of the application of the teaching learning software engineering process. Table 3 shows the elements that instantiate the sub-alphas which are part of the extension.

Table 3. Alphas, sub-alphas and instances in CSEL

Alpha	*Sub-alpha*	*Instance in CSEL*
Work	Learning activity	Task assignment Task development Sending artifacts for inspection Artifacts inspection
Way-of-working	Teaching Strategy	Communication management Artifacts management Planning management Progress management

Source: authors

3. Representing activity spaces and activities of CSEL practice: the activities proposed by CSEL integrate activity spaces that are part of the Semat kernel. This integration allows defining, monitoring and evaluating the process proposed by CSEL as practice for teaching Software Engineering. Relations between CSEL practice, activity spaces, activities and work products are shown in figure 19.

4. Representing competencies of CSEL practice: equired competencies to perform efficiently CSEL practice are shown in figure 19.

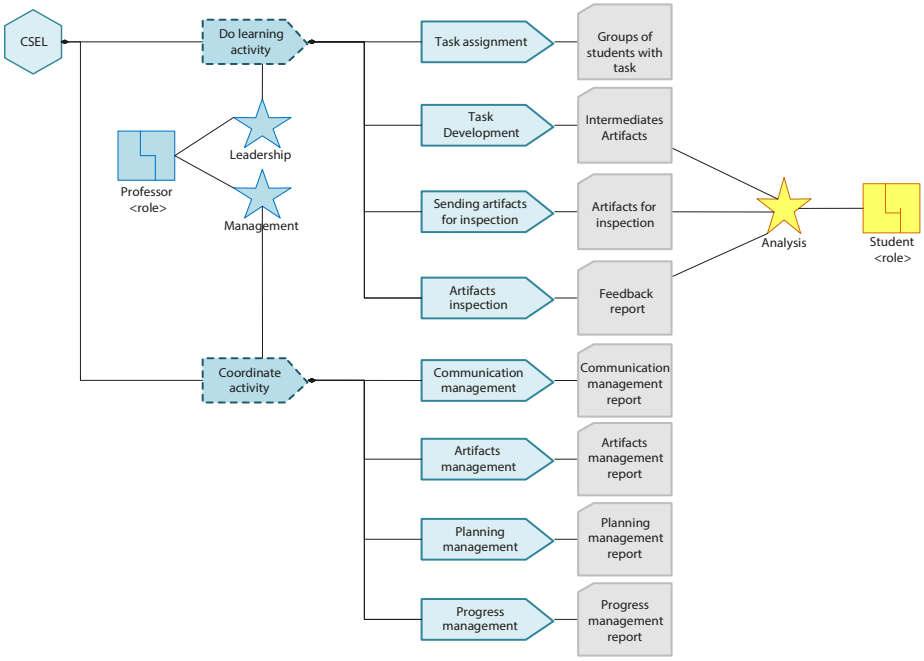

Figure 19. Activity spaces, activities, work product, competencies and roles of CSEL practice

Source: authors

5. CONCLUSIONS

This paper presents a Semat Kernel Extension proposal for representing PLTSE, for the discussion of the academic community. Such extension includes a set of main concepts and relationships in this domain identified from the analysis of a set of software engineering teaching experiences. This work shows that constructs and propositions of the Semat kernel are sufficient in general, allowing representation of a new domain with the addition of a few extension elements.

In addition, the authors propose to study the structure of Semat kernel as a general framework for representation of practices and methods from other discipline domains outside software engineering domain, like PTLSE. The results demonstrate the scalability, extensibility and ease-of-use of Essence for representation, usage and improvement of practices and methods of other disciplines. Accordingly, we propose to create a new kernel designed specifically for PTLSE with a similar Semat kernel structure.

As a future work, we propose to analyze the "Competency" construct, which in the case of PTLSE, corresponds with abilities to promote in software engineering students, finding mechanisms to measure progress of such competencies through the states of the sub-alpha "learning goal" of defining indicators for monitoring the students' competencies acquisition like rubrics.

REFERENCES

[1] I. Sommerville, *Software Engineering* (9 ed.), New York: Addison-Wesley, 2010.

[2] S. Olson and D. G. Riordan, "Engage to Excel: Producing One Million Additional College Graduates with Degrees in Science, Technology, Engineering, and Mathematics. Report to the President.," *Exec. Off. Pres.*, 2012.

[3] M. Qureshi, M. Asim, and M. Nadeem, "A new teaching model for the subject of software project management," *arXiv Prepr. arXiv*, vol. 22, no.1, pp. 295-303, 2012.

[4] F. Ma, W. Li, J. Si, and L. Wang, "Theoretical Construction of Prototype Teaching Model," in *2010 Second International Workshop on Education Technology and Computer Science*, 2010, vol. 3, pp. 802-805.

[5] C. N. Bull and J. Whittle, "Supporting Reflective Practice in Software Engineering Education through a Studio-Based Approach," *IEEE Softw.*, vol. 31, no. 4, pp. 44-50, Jul. 2014.

[6] C. Moller, G. Reina, M. Burch, and D. Weiskopf, "Large-Scale Visualization Projects for Teaching Software Engineering," *IEEE Comput. Graph. Appl.*, vol. 32, no. 4, pp. 14-19, Jul. 2012.

[7] A. Dorling and F. McCaffery, "The Gamification of SPICE," in *Software Process Improvement and Capability Determination*, Berlin: Springer Berlin Heidelberg, 2012, pp. 295–301.

[8] Object Management Group, "Kernel and Language for Software Engineering Methods (Essence)" Object Management Group, Massachusets, Document Number formal/15-12-03, Nov, 2014. [online]. Available: http://www.omg.org/spec/Essence/1.0

[9] IEEE Computer Society, P. Bourque, and R. E. Fairley, Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0. Los Alamitos, CA: *IEEE Computer Society Press*, 2014.

[10] R. J., LeBlanc, A. Sobel, J. L., Díaz-Herrera, and T. B. Hilburn. *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.* New York, NY: IEEE Computer Society, 2004.

[11] D. Madigan, and M. Print, "IEEE 1547 and 2030 Standards for Distributed Energy Resources Interconnection and Interoperability with the Electricity Grid," Agile construction initiative, 2014. Available: http://www.bath.ac.uk/management/larg_agile/publications/pdf/cnu97-38.pdf

[12] T. Basso, "IEEE 1547 and 2030 standards for distributed energy resources interconnection and interoperability with the electricity grid," National Renewable Energy Laboratory - U.S. Department of Energy Office of Energy Efficiency & Renewable Energy, United States, Technical Report NREL/TP-5D00-63157, 2014 [online]. Available: http://www.nrel.gov/docs/fy15osti/63157.pdf

[13] Government of Alberta, "Best Practices for the Assessment and Control of Chemical Hazards," vol. 3, *Best Practices Guidelines for Occupational Health and Safety in the Healthcare Industry*, Alberta: Government of Alberta, 2010. Available: https://work.alberta.ca/documents/WHS-PUB_bp011.pdf

[14] V. Dhole, R. Beck, D. Tremblay, D. Ajikutira, and S. Mullick, "Transformation of Process Engineering – Innovations and Best Practices," Aspen Technology Inc., Massachusets, White Paper 11-1195-0812 2012 [Online]. Available: https://www.aspentech.com/Transformation_of_Process_Engineering.pdf

[15] Software Engineering Institute, "CMMI for Development, Version 1.3," Carnegie Mellon Institute, Pittsburgh, Pennsylvania, Nov, 2010 [Online]. Available: https://cmmiinstitute.com/resources/cmmi-development-version-13

[16] Object Management Group, "Kernel and Language for Software Engineering Methods (Essence)-Version 1.1," Object Management Group, Massachusets, Document Number ptc/15-05-13, Dec, 2015. [online]. Available: https://www.omg.org/spec/Essence/1.1

[17] P. Grossman, C. Compton, D. Igra, M. Ronfeldt, E. Shahan, and P. Williamson, "Teaching practice: A cross-professional perspective," *Teachers College Record*, vol. 111, no. 9, pp. 2055-2100, 2009.

[18] P. E. G. Férez, "Un acercamiento al trabajo colaborativo," *Revista Iberoamericana de Educación*, vol. 36, no. 7, pp. 1-14, 2005.

[19] R. M. Harden, "Learning outcomes and instructional objectives: is there a difference?," *Medical teacher*, vol. 24, no. 2, pp. 151-155, 2002.

[20] A. García, and M. Daneri, "Investigación e innovación en el conocimiento educativo actual," En R. I. Roig y J.E. Blasco (Coord.), 2008.

[21] H. Chen and K. Damevski, "A teaching model for development of sensor-driven mobile applications," in *Proceedings of the 2014 conference on Innovation & technology in computer science education*, 2014, pp. 147-152.

[22] M. L. I. Forneiro, "Observación y evaluación del ambiente de aprendizaje en educación infantil: dimensiones y variables a considerar," *Revista Iberoamericana de educación*, no. 47, pp. 49-70, 2008.

[23] A. M. Vivar Quintana, A. B. González Rogado, A. B. Ramos Gavilán, I. R. Martín, M. Ascensión, R. Esteban, T. A. Zorrila, and J. F. Martín Izard, "Application of rubric in learning assessment: a proposal of application for engineering students," in *Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality*, 2013, pp. 441–446.

[24] M. Spellings, *A test of leadership: Charting the future of US higher education*. Washington DC: US Department of Education, 2006.

[25] D. Golden, "Colleges, accreditors seek better ways to measure learning," *Wall Street Journal*, pp. 738027–298, 2006.

[26] Merriam Webster, "Definition of conceptualization." [Online]. Available: https://www.merriam-webster.com/dictionary/conceptualization.

[27] K. J. Stol and B. Fitzgerald, "Uncovering theories in software engineering," in *2nd Semat Workshop on a General Theory of Software Engineering (GTSE), 2013*, San Francisco, California, United States, 2013, pp. 5–14.

[28] C. M. Zapata Jaramillo, A. Gelbukh, and F. Arango Isaza, "Pre-conceptual Schema: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation," in SpringerLink, 2006, pp. 27–37, [Online]. Available: https://nlp.cic.ipn.mx/Publications/2006/MICAI-2006-Preconceptual.pdf.

[29] C. M. Zapata, G. L. Giraldo, and S. Londoño, "Esquemas preconceptuales ejecutables," *Avances en Sistemas e Informática*, vol. 8, no. 1, pp. 15–24, 2011.

[30] D. I. K. Sjøberg, T. Dybå, B. C. Anda, and J. E. Hannay, "Building theories in software engineering," in *Guide to advanced empirical software engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. London: Springer, 2008, pp. 312–336.

[31] M. Ekstedt, "An empirical approach to a general theory of software (engineering)," in *2nd Semat Workshop on a General Theory of Software Engineering (GTSE), 2013*, San Francisco, California, United States, 2013, pp. 23–26.

[32] Semat.org, "What is it and why should you care? - Semat." [Online]. Available: http://semat.org/what-is-it-and-why-should-you-care-.

[33] B. Elvesæter, G. Benguria, and S. Ilieva, "A comparison of the Essence 1.0 and SPEM 2.0 specifications for software engineering methods," in *Proceedings of the Third Workshop on Process-Based Approaches for Model-Driven Engineering*, New York, NY, 2013, p. 2.

[34] Object Management Group, "Software & Systems Process Engineering Meta-Model Specification" Object Management Group, Massachusets, Document Number formal/08-04-01, Apr, 2008. [online]. Available: https://www.omg.org/spec/SPEM/2.0/PDF.

[35] ISO, "ISO 9000 - Quality management," ISO 9000 - Quality management, 2015. [Online]. Available: http://www.iso.org/iso/home/standards/management-standards/iso_9000.htm.

[36] I. Jacobson, P.-W. Ng, P. McMahon, I. Spence, and S. Lidman, "The Essence of Software Engineering: The Semat Kernel," *Queue*, vol. 10, no. 10, pp. 40:40–40:51, Oct. 2012.

[37] I. Jacobson, P.-W. Ng, P. E. McMahon, I. Spence, and S. Lidman, *The Essence of Software Engineering: Applying the Semat Kernel*, Upper Saddle River, NJ: Addison-Wesley Educational Publishers Inc, 2013.

[38] C. M. Zapata-Jaramillo, "An executable pre-conceptual schema for a software engineering general theory," in *Software Engineering: Methods, Modeling*, vol. 3, C. M. Zapata-Jaramillo and L. F. Castro-Rojas, Eds. Medellín: Centro Editorial de la Facultad de Minas, 2014, pp. 3–7.

[39] C. Zapata, L. Castro, H. Shihong, and P.-W. Ng, "Preface," in *Software Engineering: Methods, Modeling, and Teaching*, vol. 3, C. M. Zapata-Jaramillo and L. F. Castro-Rojas, Eds. Medellín: Centro Editorial de la Facultad de Minas, 2014, pp. v–ix.

[40] T. Guowei, G. Lingling, F. Yu, L. Jinghui, and Z. Wanping, G. Lee, Eds., "Research and Practice on 'Triple-driven' Based Software Development Practical Teaching System," in International Conference on Future Computer Supported Education, Fraser Place Central - Seoul, IERI Procedia, Aug 22- 23, 2012, Vol. 2, pp 18-23.

[41] D. I. K. Sjøberg, T. Dybå, B. C. Anda, and J. E. Hannay, "Building theories in software engineering," in *Guide to advanced empirical software engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. London: Springer, 2008, pp. 312–336.

[42] M. Ekstedt, "An empirical approach to a general theory of software (engineering)," in *2nd Semat Workshop on a General Theory of Software Engineering (GTSE), 2013*, San Francisco, California, United States, 2013, pp. 23–26.

[43] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[44] R. Sánchez-Dams, A. Barón-Salazar, and M. C. Gómez-Álvarez, "An Extension of the Semat Kernel for Representing Teaching and Learning Practices about Embedded Systems," in *4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, Ciudad de México, 2016, pp. 39–46.

[45] A. Hazeyama, "Collaborative Software Engineering Learning Environment Associating Artifacts Management with Communication Support," *IIAI 3rd International Conference on Advanced Applied Informatics (IIAIAAI), 2014*, Kitakyushu, 2014, pp. 592-596.